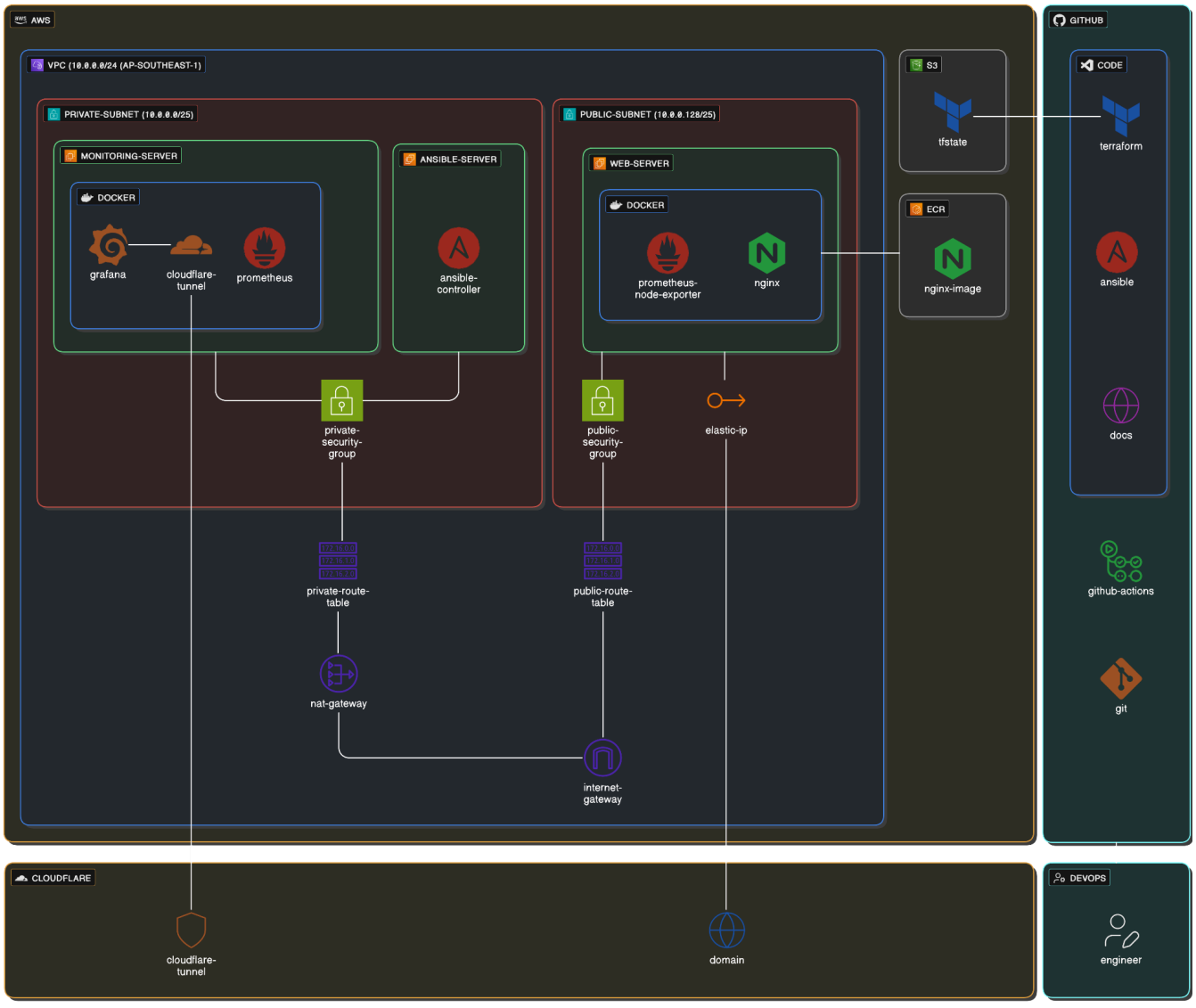
DevOps Bootcamp Final Project 2025



Project Architecture Diagram

Project Name : Trust Me, I'm a DevOps Engineer

1. Introduction

This final project is a graded capstone assignment for the DevOps Bootcamp.

The purpose of this project is to assess your ability to design, provision, configure, deploy, monitor, and document a complete DevOps-based system using industry-standard tools and practices.

Successful completion of this project is mandatory in order to:

- Pass the bootcamp
- Receive the DevOps Bootcamp Certificate

2. Project Objectives

By completing this project, you will demonstrate the ability to:

1. Provision AWS infrastructure using Infrastructure as Code (Terraform)
2. Apply configuration management using Ansible
3. Deploy a containerized web application
4. Implement monitoring using Prometheus and Grafana
5. Use DevOps tools, automation and best practices
6. Publish technical documentation using GitHub Pages

3. Scope, Rules & Important Notes

Please read this section carefully before starting.

- All infrastructure must be provisioned using Terraform
- All servers must be configured using Ansible
- You must follow the instructions exactly as stated, unless otherwise specified
- The project focuses on infrastructure and deployment, not application development
- The web application source code will be provided by the instructor
- CI/CD usage is basic and limited to documentation updates but bonus if you can push the container image to ECR using GitHub Action
- Security hardening is optional but considered a bonus

4. Infrastructure Requirements (Terraform)

4.1 Terraform State Storage

Create an Amazon S3 bucket to store the Terraform state file:

- Bucket Name: devops-bootcamp-terraform-yourname
- Region: ap-southeast-1

4.2 Network Architecture

Provision the following AWS resources in ap-southeast-1 by using Terraform :

- VPC : 10.0.0.0/24 (name : devops-vpc)
- Subnets:
    - Public Subnet: 10.0.0.0/25 (name : devops-public-subnet)
    - Private Subnet: 10.0.0.128/25 (name : devops-private-subnet)
- Route Tables:
    - 1 for Public Subnet (name : devops-public-route)
    - 1 for Private Subnet (name : devops-private-route)

- Internet Gateway: Required for public subnet (name : devops-igw)
- NAT Gateway: Required for private subnet (name : devops-ngw)

4.3 Security Groups

Create two security groups for:

a) Web Server Security Group (name : devops-public-sg)

- Port 80: Allow from any IP
- Port 9100: Allow from Monitoring Server IP (For Prometheus Node Exporter)
- Port 22: Allow from VPC subnet only

b) Ansible Controller & Monitoring Server Security Group
(name : devops-private-sg)

- Port 22: Allow from VPC subnet only

4.4 EC2 Instances

Provision three EC2 instances using Terraform with the specifications below.

a) Server 1 – Web Server

- AMI : Ubuntu 24.04
- Subnet: Public
- Instance Type: t3.micro
- Disk: Default
- Private IP: 10.0.0.5
- Elastic IP: Must be allocated and associated
- Security group : devops-public-sg

b) Server 2 – Ansible Controller

- AMI : Ubuntu 24.04
- Subnet: Private
- Instance Type: t3.micro
- Disk: Default
- Private IP: 10.0.0.135
- Must NOT have a public IP
- Security group : devops-private-sg

c) Server 3 – Monitoring Server

- AMI : Ubuntu 24.04
- Subnet: Private
- Instance Type: t3.micro
- Disk: Default
- Private IP: 10.0.0.136
- Must NOT have a public IP
- Access to the monitoring tool must be published via Cloudflare Tunnel only
- Security group : devops-private-sg

4.5 Container Registry

Create an Amazon Elastic Container Registry (ECR):

- Repository Name:
  devops-bootcamp/final-project-yourname

---

5. Application & Configuration Management (Ansible)

All configurations must be executed from the Ansible Controller server.

5.1 Application: my-devops-project

- The my-devops-project website source code can be cloned from : https://github.com/Infratify/lab-final-project
- Students must use the provided source code as the web application.
- Application code modification is not required and not graded
- The focus is on:
  - Dockerizing the application
  - Deploying it on the web server
  - Exposing it via the assigned domain

5.2 Web Server Configuration

Using Ansible:

- Install Docker Engine
- Deploy the my-devops-project web application as a Docker container
- Ensure the application is accessible via HTTP (port 80)

5.3 Monitoring Server Configuration

By Using Ansible:

- Install Docker on the monitoring server
- Deploy:
    - Prometheus
    - Grafana
- Configure Prometheus to collect metrics from the web server such as CPU, RAM and DISK

---

6. Monitoring Requirements

6.1 Prometheus Metrics

Prometheus must monitor the web server for:

- CPU usage
- Memory usage
- Disk usage

6.2 Grafana Dashboards

Grafana must visualize:

- CPU usage
- Memory usage
- Disk usage

7. Access & Connectivity

7.1 AWS Systems Manager (SSM)

- SSM must be enabled on all servers
- Used for manual access and troubleshooting

7.2 Ansible Connectivity

- Ansible may use SSH
- Using SSM for Ansible is not required

8. Domain & Cloudflare Configuration

1. You will need a valid registered domain name, hosted at Cloudflare.
2. Use Cloudflare to manage the DNS records
3. Set Cloudflare SSL mode to Flexible
4. Configure the following subdomains:
    a. Web Application
        i. web.yourdomain.com
        ii. Point to Web Server Elastic IP
    b. Monitoring (Grafana)
        i. monitoring.yourdomain.com
        ii. Access Grafana via Cloudflare Tunnel
        iii. Grafana shows the least metric of CPU, RAM and DISK of the web server.
        iv. The monitoring server must not be publicly exposed

9. CI/CD Requirements

- Use GitHub Actions to update the documentation website on Github Pages
- CI/CD for application deployment such as push Docker Image to ECR and pull and run on the web server is a bonus!

10. GitHub Repository Structure

Create one public GitHub repository:

- Repository Name: devops-bootcamp-project

Required Structure

devops-bootcamp-project/
├── terraform/ - store all the terraform codes
├── ansible/ - store all the ansible playbook
└── README.md - The documentation page, served by Github Pages

Make sure the repository is set to public

11. Documentation (GitHub Pages)

Your documentation must be published using GitHub Pages.

Documentation URL
https://username.github.io/devops-bootcamp-project

Mandatory Documentation Content

Your documentation page must include:

1. Web application URL
   web.yourdomain.com
2. Monitoring URL
   monitoring.yourdomain.com
3. GitHub repository URL

Documentation Guidelines

- Final URLs are compulsory
- Step-by-step setup guide is recommended
- Screenshots are optional but encouraged

12. Submission Instructions

Submit the following via the Infratify platform:

1. GitHub Pages documentation URL
2. Grafana login credentials

13. Bonus (Optional)

Additional marks may be awarded for:

- Secure infrastructure design
- Least-privilege IAM usage
- Full Infrastructure as Code coverage
- GitHub Actions to:
    - Build Docker image
    - Push image to ECR
    - Pull and deploy using a self-hosted GitHub Runner (or you can use Ansible)

14. Completion Criteria

You will be considered successful if:

- All infrastructure is provisioned correctly
- Web application is accessible via the assigned domain
- Monitoring dashboards are functional
- Documentation is publicly accessible
- All requirements are met as specified

📘 Appendix A: How to start

This section provides a **guideline step-by-step order** to complete the DevOps Bootcamp Final Project successfully. Students are encouraged to follow this order to avoid dependency issues and rework.

STEP 1: Preparation & Prerequisites

1. Ensure you have:
    - An active AWS account
    - A registered domain name added to Cloudflare
    - A GitHub account

2. Create a **public GitHub repository** named: `devops-bootcamp-project`
3. Prepare local tools:
    - Terraform
    - AWS CLI
    - Git

STEP 2: Terraform – Infrastructure Provisioning

4. Create an **S3 bucket** for Terraform state storage.
5. Configure Terraform backend to use the S3 bucket.
6. Provision networking resources:
    - VPC
    - Public and private subnets
    - Route tables
    - Internet Gateway
    - NAT Gateway
7. Create required **security groups**.
8. Provision EC2 instances:
    - Web Server (public subnet + Elastic IP)
    - Ansible Controller (private subnet)
    - Monitoring Server (private subnet)

9. Enable **AWS Systems Manager (SSM)** access on all servers.
10. Create an **ECR repository**.

✅ At this stage, all infrastructure should be running successfully.

STEP 3: Ansible – Configuration Management

11. Access the Ansible Controller (via SSM or bastion access).
12. Configure Ansible inventory using **private IP addresses**.
13. Use Ansible to Install Docker on all relevant servers

STEP 4: Build Docker Image and Deploy

14. Clone final project lab repo and build docker image : https://github.com/Infratify/lab-final-project
15. Push to ECR
16. Deploy it on web server
17. Verify by access the web server public IP and you should get the web page

STEP 5: Monitoring & Observability

15. Deploy Prometheus on the monitoring server - Use Docker!
16. Configure the Prometheus to collect data from the web server resources such as CPU usage, RAM usage and DISK usage.
16. Deploy Grafana on the monitoring server - Use Docker!
17. Configure Grafana dashboards to visualize the metrics from Prometheus
18. Confirm metrics are updating in real time.

STEP 6: Domain & Cloudflare

18. Configure Cloudflare DNS. Point `web.yourdomain.com` to the Web Server Elastic IP

19. Create a **Cloudflare Tunnel**:
- Expose Grafana securely via monitoring.yourdomain.com
- Ensure Monitoring Server has **no public access**

20. Verify:
- Web application accessible via domain
- Grafana accessible only via Cloudflare Tunnel

STEP 7: Documentation & CI/CD

21. Write project documentation in `README.md`.
22. Enable **GitHub Pages** for the repository.
23. Set up **GitHub Actions** to auto-deploy documentation updates.
24. (Optional Bonus) Implement CI/CD to:
- Build Docker image
- Push to ECR
- Pull and deploy to Web Server

STEP 8: Final Validation & Submission

25. Validate all completion criteria.

26. Submit below into the Infratify platform::

- GitHub Pages documentation URL
- Grafana login credentials

📊 Appendix B: Grading Rubric

The final project is graded based on the following rubric.
All mandatory sections **must be completed** to pass.

1. Infrastructure Provisioning (Terraform) — 30%

Criteria	Marks
Terraform backend (S3) configured	5%
VPC, subnets, routing, gateways	10%
Security groups correctly implemented	5%
EC2 instances provisioned correctly	10%

2. Configuration Management (Ansible) — 25%

Criteria	Marks
Ansible controller setup	5%
Docker installed via Ansible	5%
Web application deployed via container	10%
Configuration reproducibility	5%

3. Monitoring & Observability — 20%

Criteria	Marks
Prometheus configured correctly	8%
Grafana dashboards functional	8%
Metrics collected from Web Server	4%

4. Domain & Secure Access — 15%

Criteria	Marks
Web app accessible via domain	5%
Cloudflare Tunnel implemented	7%
Monitoring server not publicly exposed	3%

5. Documentation (GitHub Pages) — 10%

Criteria	Marks
GitHub Pages enabled	3%
URLs clearly documented	4%
Clear, structured explanation	3%

🎯 Bonus Marks (Up to +10%)

Bonus Feature	Marks
Full IaC coverage	+3%
CI/CD: Build & push to ECR	+3%
CI/CD: Deploy image to web server	+4%

✅ Passing Criteria

- Minimum **60% total score**
- All **mandatory sections completed**
- Working URLs and dashboards